IN DIESEM KAPITEL...

- Umzug ins Ungewisse
- Orientierung
- Nette Viertel
- Politische Entscheidungsprozesse

Kapitel 1
Python – eine Hochglanzbroschüre

Wenn Sie schon mal umgezogen sind, dann kennen Sie vielleicht das mulmige Gefühl, das einen beim Gedanken an den neuen Wohnort beschleicht. Die Straßennamen klingen fremd, Sie kennen die Nachbarn noch nicht und sind mit deren Gepflogenheiten nicht vertraut. Eine neue Programmiersprache zu lernen, fühlt sich fast genauso an, aber vielleicht hilft Ihnen diese Broschüre, sich erst mit dem Gedanken an den neuen Wohnort und dann an das schmucke Dörfchen selbst anzufreunden.

Erinnern Sie sich nur an die Gründe, warum Sie gerade umziehen! Vielleicht ist es einer der folgenden?

- ✔ Überteuerte Miete und/oder Betriebskosten.
- ✔ Überfällige Reparaturen oder Schäden an der Wohnung.
- ✔ Probleme mit den Eltern, Nachbarn oder dem Vermieter.
- ✔ Änderung der Lebensumstände – ein neuer Job oder eine neue Beziehung?
- ✔ Sie fühlen sich in Ihrer alten Wohnung unwohl.

Es ist egal, was genau Sie vorher programmiert haben, es gibt viele gute Gründe, mit der Programmierung nach Python umzuziehen – oder Python zu wählen, falls es Ihre erste eigene Wohnung – Pardon: Programmiersprache – ist. Wäre Python eine neue Wohnung, so sprächen die folgenden Punkte dafür:

- ✔ Niedrige Betriebskosten.
- ✔ Modernisierter Altbau, der sehr gut instand gehalten wird.

- ✔ Supernette Nachbarn und der Vermieter ist ein Menschenfreund.
- ✔ Egal welchen Job Sie in Zukunft annehmen werden, Sie können fast immer in Python wohnen bleiben.
- ✔ Sie werden sich einfach sehr wohl fühlen.

Sobald das Einzugsdatum feststeht (jetzt!), müssen Sie erst mal Ihre neue Wohnung einrichten und daher fängt auch dieses Buch mit einem kurzen Überblick und einer Installationsanleitung an. Danach sollten Sie Ihre neue Wohnung genauer kennenlernen, daher folgen einige Buchteile über die verschiedensten Aspekte der Syntax dieser Sprache. Später sollten Sie mal einen Stadtrundgang wagen, um zu gucken, was Sie von Ihrer Wohnung aus alles erreichen können, dementsprechend behandeln die letzten Buchteile praktische Anwendungen.

Was Python so interessant macht

Wenn Sie sich in Ihrem neuen Wohnort umsehen, so werden Sie viele schöne Stadtteile entdecken:

- ✔ Python bietet eine reichhaltige und umfassende Standardbibliothek.
- ✔ Flexible und starke Datenstrukturen (zum Beispiel Listen, Dictionarys und Mengen).
- ✔ Interessante Bauwerke, die von anderen Kulturen geprägt wurden (z.B. Comprehensions, Iteratoren, Dekoratoren und Deskriptoren).

Eine Python-Installation von der Stange enthält Module für die Verarbeitung von Texten, Kalendern, Zeit und Datumsangaben. Mit von der Partie sind auch Datei- und Pfadoperationen. Ohne Umstände können Sie bereits Zip-Dateien schreiben und lesen. Außerdem kann Python von Haus aus viele gängige Datenformate verarbeiten, etwa HTML, XML, CSV und JSON. Für die Programmierung von netzwerkfähigen Anwendungen gibt es Module für IP-Adressen und verschiedene Protokolle. Obendrein kann man mit Python Software-Tests gestalten und GUIs entwerfen – und das alles, ohne außer Python noch andere Tools zu installieren. So ist man für den Alltag als Programmiererin oder Admin gewappnet.

In Python legt sich eleganter Code beim Schreiben von alleine unter die Hände, denn für die wichtigsten Anwendungsfälle sind keinerlei Umwege erforderlich. In Java und C# etwa muss man für die einfachsten Operationen immer erst irgendwelche Module nachladen, aber in Python sind die wichtigsten Funktionen einfach Teil der Sprache und haben eine prominente Syntax. Dadurch hat man die wichtigsten Algorithmen und Datenstrukturen stets griffbereit. So etwa erzeugt man eine Liste und gibt sie aus:

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(numbers)
```

Über die Jahre sind diverse Konzepte in die Sprache eingewandert, die sich in anderen Programmiersprachen bewährt haben. Prominentes Beispiel etwas sind Comprehensions, mit

denen Sie Datentransformationen kurz und knackig beschreiben können. Das folgende Programm etwa gibt den Text »jhnns« aus.

```
name = "johannes"
consonants = [c for c in name if c not in "aeiou"]
print(consonants)
```

Aber das sind erst mal nur syntaktische Oberflächlichkeiten. Mit etwas Erfahrung werden Sie die konzeptionelle Qualität von Python schätzen lernen. Python hat sich über die Jahre stets von den besten Ideen anderer Programmiersprachen inspirieren lassen und vereint Aspekte von funktionaler und objektorientierter Programmierung sowie Konzepte aus C, C++, Lisp, Self, ML, Haskell und Perl.

Dunkle Flecken – was an Python manchmal nicht so schön ist

An jedem Wohnort gibt es auch dunkle Ecken, die man eher meidet – nicht unbedingt No-go-Zonen, wo man sofort abgestochen wird, während man einem rechtschaffenen Tauschgeschäft nachgeht (z.B. Drogen gegen Geld), aber es gibt doch Orte, die man als Problembezirk bezeichnen könnte und die nicht so gerne besucht werden. Aber auch da wohnen Leute und berichten manchmal Gutes – wie es ja auch Leute geben soll, die gerne in Mannheim wohnen.

- ✔ GUIs sind in Python eher so mittelmäßig gut zu implementieren.
- ✔ Python ist nicht über die Maßen schnell.
- ✔ Parallele Programmierung braucht viel technisches Wissen.
- ✔ Python 2.

Gehen wir auf die Punkte etwas näher ein. GUIs sind zum Beispiel so ein Thema. Python ist zwar plattformunabhängig und Programme lassen sich meist sowohl auf Windows als auch unter Linux und macOS ausführen, sobald man aber grafische Oberflächen programmieren möchte, wird es hakelig. Von Haus aus bringt Python Tkinter mit, was aber etwas altbacken daherkommt. Für moderne GUIs, gar mit Touch-Funktionalität, muss man meist ein Toolkit verwenden und damit verlässt man häufig die elegante Python-Welt der klaren Konzepte und schlichten Syntax. Man kommt zum Ziel, aber das ist dann schon recht anspruchsvoll.

Zugegebenermaßen ist Python auch nicht die schnellste Programmiersprache der Welt – allerdings kommt es hier darauf an, wie man die Sache betrachtet. Python-Code wird durch einen Interpreter ausgeführt und nicht vor oder während der Ausführung durch Kompilieren in Maschinencode übersetzt. Dadurch ist das Ganze natürlich viel langsamer als ein Programm, dass Sie in C oder Rust schreiben. So gesehen ist aber nicht die Sprache selbst langsam, sondern die Ausführungsgeschwindigkeit wird durch den verwendeten Interpreter limitiert. Es gibt alternative Interpreter außer dem hauseigenen (der sich *CPython* schimpft, weil er selbst in C geschrieben wurde), die Ihre Situation verbessern können, wie zum Beispiel den des PyPy-Projekts. Wenn es allerdings hart auf hart kommt und Sie das letzte

Quäntchen an Leistung herauskitzeln möchten, dann sollten Sie aufwändigere Teile des Programms in einer kompilierten Sprache schreiben. Für Computerspiele etwa lohnt es sich, Routinen für lineare Algebra und Gewurschdel auf der Grafikkarte in eine schnelle C++-Bibliothek auszulagern, aber die Logik des Spiels läuft weiter in Python.

Schnelligkeit gewinnt man in Python an einer anderen Stelle: Durch die Klarheit und Ausdrucksstärke der Sprache kommt man beim Coden sehr schnell zum Ziel, schreibt tendenziell besseren Code und ist flexibel. Ihr Programm ist daher schneller programmiert und fehlerärmer.

Grundsätzlich lassen sich bestimmte Programme auch beschleunigen, indem man sie parallelisiert und damit moderne Prozessorarchitekturen nutzt. Allerdings gibt es in Python eine Sperre, um die man herumarbeiten muss, wenn man mit Threads arbeitet. Trotz all dieser Einschränkungen ist Python in der Wissenschaft sehr beliebt und es gibt viele Bibliotheken in den Bereichen Statistik, Machine Learning und Künstliche Intelligenz.

Zu guter Letzt gibt es da noch die Sache mit Python 2. Dieses Hochhausghetto soll seit Jahren abgerissen werden, aber die letzten Mieter ziehen und ziehen einfach nicht aus. Python liegt aktuell in der Version 3.12 vor. Da die Entwickler von Python regelmäßig neue Releases veröffentlichen, ist die Version sicher schon bei 3.14, wenn dieses Buch fertig ist und bei 3.15, wenn Sie sich zum Kauf entschieden haben. Früher gab es aber die Version 2, die bis zur Version 2.7 verbessert wurde. Die Einführung von Python 3 sollte die Version 2 ersetzen und dabei einige Probleme beseitigen, allerdings waren die Versionen nicht ohne Nacharbeit kompatibel zueinander, sodass die Umstellung nicht ganz reibungslos verlief und auch noch nicht ganz abgeschlossen ist. Es gibt immer noch Codebasen, die Version 2.7 verwenden. Dieses Problem wird mit der Zeit an Relevanz verlieren, aber es kann Ihnen durchaus passieren, dass Sie in einem Python-Job ein Programm von 2.7 auf 3.0 portieren müssen.

Infrastruktur und Dienstleistungen

Wenn es in Ihrem richtigen Wohnort gut läuft, dann liegt das mit Sicherheit daran, dass zwei Bereiche ordentlich zusammenarbeiten:

- ✔ Die öffentliche Hand kümmert sich um die Grundversorgung.
- ✔ Privatwirtschaftliche Betriebe übernehmen Zusatzversorgung, gehen aber manchmal auch risikohafte Investitionen ein.

So ist das auch in Python. Python sorgt stets dafür, dass der Müll abgeholt wird (buchstäblich: es gibt einen »Garbage Collector«). Die Anschlüsse für Gas, Wasser und Strom funktionieren, die Straßen sind geteert und der Bus kommt pünktlich. Wenn Sie etwas brauchen, gehen Sie einfach aufs Amt – die Sprache und die Standardbibliothek sind stets in Laufweite und der öffentliche Sektor in Python ist auch sonst sehr gut aufgestellt.

Allerdings reicht das oft nicht, denn oftmals sind die städtischen Busse überfüllt. Gut, dass es hier private Busunternehmen gibt, die einspringen.

Der »private Sektor« ist im Bezug auf Python durchaus hervorzuheben. Es gibt zig Firmen und Projekte, die Zusatzmodule für Python beisteuern, dabei sind die meisten nicht einmal gewinnorientiert. So finden Sie Lösungsmodule für alle möglichen Anwendungsfälle, inklusive sehr esoterischer Probleme oder Unternehmensanwendungen, die den Funktionsumfang der Standardbibliothek weit übersteigen.

Zum Beispiel bringt Python ein Modul für Web-Anfragen mit, die *urllib*, das aber etwas sperrig zu benutzen ist. Glücklicherweise kann man auf das Projekt *requests* zurückgreifen, das Ihnen hilft, mit Cookies, Downloads, Streaming oder seltsamen Headern umzugehen. Das gleiche gilt für wissenschaftliche Anwendungen. Python wird mit einem kleinen, aber feinen Statistikmodul ausgeliefert. Für ernstzunehmende Probleme lädt sich der Profi dann die Pakete *Numpy* und *Scipy*, die hochoptimierten Code bieten.

In Gänze hat das Ökosystem wirklich viel zu bieten und Sie werden es kaum schaffen, ein Problem zu finden, für das es *kein* Python-Modul gibt.

Haushalt und Wirtschaft

Finanziell steht die Python-Community gut da:

- ✔ Die Python Software Foundation (PSF) sammelt Spenden und unterstützt die Entwicklung der Programmiersprache.
- ✔ Python erhält finanzielle Unterstützung von verschiedenen großen Konzernen wie Microsoft, Google, Amazon oder Nvidia.

Python ist kein Produkt einer einzelnen Firma, sondern entstand als Projekt an einer Universität und ist relativ unabhängig von den wirtschaftlichen Interessen einer einzelnen Erfinderfirma. Daher brauchen Sie sich auch keine Sorgen zu machen, dass Python morgen überraschend von der Bildfläche verschwinden könnte, da Python längst nicht mehr aus dem Enterprise-Umfeld wegzudenken ist und verschiedene Firmen einiges an Ressourcen bereitstellen, um die Entwicklung immer weiter voranzutreiben.

Das wiederum lässt natürlich an der Unabhängigkeit zweifeln. Wo ein Sponsoring ist, da sind meist auch wirtschaftliche Partikularinteressen und so sind Interessenskonflikte vorprogrammiert. Im April 2021 etwa berichtete Microsoft, dass fünf Python-Kernentwickler bei Microsoft angestellt seien, nämlich Brett Cannon, Steve Dower, Guido van Rossum, Eric Snow und Barry Warsaw. Wer nun Sorge hat, dass Microsoft den Ton angibt und Python damit immer mehr in eine unsinnige Enterprise-Ecke gerät, der kann eventuell dadurch beruhigt werden, dass es sich bei Guido van Rossum um den Erfinder der Sprache höchstpersönlich handelt. Daher kann man bis auf Weiteres schon davon ausgehen, dass Python sich im Sinne des Erfinders weiterentwickelt.

Diese Diskussion wirft die Frage auf: Welche Mechanismen gibt es, um Unabhängigkeit und Gemeinwohl von Python gegenüber individuellen Interessen zu verteidigen?

Politische Entscheidungsprozesse

Python wird von einer offenen Community weiterentwickelt:

- ✔ Der Quellcode ist offen einsehbar.
- ✔ Änderungsvorschläge werden als PEPs eingereicht.
- ✔ Die Entscheidungsfindung läuft weitestgehend basisdemokratisch ab und die Mitbestimmung der Community wird großgeschrieben.
- ✔ Lange Zeit hatte der BDFL ein letztes Wort.

Die Grundlage für alle Änderungen bilden die *Python Enhancement Proposals*, kurz *PEP* genannt. Was genau ein PEP ist, wurde im ersten PEP festgehalten: `https://peps.python.org/pep-0001/`. Es handelt sich um strukturierte Dokumente, die neue Features und Veränderungen der Sprache oder der organisatorischen Prozesse beschreiben und motivieren sollen. Welche Änderungsvorschläge durchgehen, entscheidet eine Art *Gemeinderat*, das *Steering Council*, ein gewähltes Komitee aus fünf Personen. Wie es sich zusammensetzt und wie aus einem Vorschlag eine Veränderung wird, beschreibt PEP 13: `https://peps.python.org/pep-0013/`.

Nicht immer wurden die Geschicke durch das Steering Council gelenkt – für lange Zeit hatte Guido van Rossum das letzte Wort, wenn es darum ging, umzusetzende Verbesserungsvorschläge anzunehmen und die Python-Geschicke zu lenken. Er hatte dadurch den Titel »Wohlwollender Diktator auf Lebenszeit« (BDFL, auf Englisch: Benevolent Dictator for Life) inne. Lange ging das gut, doch im Jahr 2018 kriselte es am Hofe des Python-Königs. Van Rossum hatte höchstpersönlich einen Vorschlag eingebracht (PEP 572), der damals sehr gemischt aufgenommen wurde, was dazu führte, dass er eine unbefristete Auszeit vom Amt des BDFL nahm. Glücklicherweise gab es damals keine Schlammschlacht und Guido van Rossum trägt heute noch immer zu Python bei – der Änderungsprozess von damals wurde in PEP 8000 dokumentiert: `https://peps.python.org/pep-8000/`.

Philosophie

Beim Umzug sollte man stets beachten, dass man mit den Leuten am Zielort klarkommt. Um herauszufinden, wie die Leute dort so drauf sind, kann man sich am Leitspruch des Zielorts orientieren, so er denn einen hat – falls nicht, tut es auch der Wahlspruch des Bundeslands.

In Rheinland-Pfalz zum Beispiel slogant man: »Wir machen's einfach«, und schon ist geklärt, wie man bei aller Rationalität bei so etwas wie dem Nutri-Score rauskommt, der seit einigen Jahren die Packungen von Tiefkühlpizzen, Paniermehl und Motoröl ziert. Der stammte von der damaligen, aus RLP stammenden Ernährungs- und Landwirtschaftsministerin, die es wohl »einfach machte«, ohne sich von irgendwelchen Konsequenzen den Spaß verderben zu lassen.

Was Python für eine Umgebung ist, erfährt man am Besten aus dem *Zen of Python*, das Tim Peters einst formulierte. Diese zwanzig Leitsätze sind so grundlegend für die Python-Denke, dass sie auf Seite 0 dieses Buches abgedruckt sind, direkt nach dem Buchcover. Sie wurden auch als PEP 20 festgehalten: `https://peps.python.org/pep-0020/`.

Python verspricht einen offenen, ehrlichen Umgang und das gilt sowohl für Rückmeldungen des Interpreters an die Programmiererin als auch für die Programmierenden untereinander.

Willkommen

Nun, sicherlich haben Sie jetzt Lust auf mehr bekommen. Python ist definitiv eine behagliche Umgebung und eine wunderbare Wahlheimat. Und selbst wenn Sie nicht immer hier wohnen bleiben, können Sie gerne regelmäßig zu Besuch kommen. Sogar wenn Sie dauerhaft in einer Java-Ruinenstadt oder einer C++-Industriebrache schmachten müssen, so können Sie gerne Ihre Urlaube in Python verbringen. Der Zuzug ist nach wie vor hoch und erinnert mehr an Berlin als an Mannheim. Ob der Trend so bleibt? Es ist auf jeden Fall genug Platz für alle da.

Wenn Sie bereit für den Umzug sind, können Sie direkt weiterlesen, um mit dem Einrichten der Wohnung zu beginnen – der Installation von Python auf Ihrem Rechner. Danach folgt eine kleine Erkundungstour durch die Nachbarschaft mit dem REPL-Fahrrad. Außerdem lernen Sie ein paar Einwohner kennen, die Ihnen zukünftig weiterhelfen, wenn Sie mal nicht weiter wissen.